OULUN SEUDUN
AMMATTIKORKEAKOULU

Juho Pekkala

MOBIILIPELINKEHITYS ALOITTAVALLE PELIYRITYKSELLE UNITY3D-YMPÄRISTÖSSÄ

MOBIILIPELINKEHITYS ALOITTAVALLE PELIYRITYKSELLE UNITY3D-YMPÄRISTÖSSÄ

Juho Pekkala
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Juho Pekkala
Opinnäytetyön nimi: Mobiilipelinkehitys aloittavalle peliyritykselle Unity3D-ympäristössä
Työn ohjaaja: Markku Rahikainen
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 44

Työn tavoitteena oli tehdä selvitys tekijän alulla olevan peliyrityksen tarvittaville menetelmille mobiilipelien kehityksessä, tutustua pelien tekemisen saloihin sekä saada vaiheistettu kokonaissuunnitelma siitä, kuinka voidaan aloittaa mobiilipelinkehitys Unity3D-ympäristössä.

Työn tarkoituksena oli tehdä tutkimus mobiilipelinkehityksen edellytyksistä Unity3D-ympäristössä. Tutkimuksessa perehdyttiin perinteisiin ohjelmistokehityksen osa-alueisiin, kuten työhön sopivaan ohjelmistokehitysmenetelmään, projekti- ja versionhallintaan ja samalla pelinkehityksen vaiheisiin sekä laitteisto- ja ohjelmistovaatimuksiin. Työssä pyritään käyttämään hyödyksi jo omistettuja ohjelmistoja ja mahdollisimman tehokkaasti avoimen lähdekoodin työkaluja.

Työssä saavutettiin erinomainen kartoitus, mitä aloittava peliyritys mobiilinpelinkehityksen aloittamiseksi Unity3D-ympäristössä tarvitsee.
Opinnäytteen valmistuttua tekijä aloittaa sivutoimisen yrittämisen mobiilipelien kehittämisen parissa hyödyntäen opinnäytteen luomaa pohjaa.

Asiasanat: mobiilipelit, mobiilipelinkehitys, peliyritys, Unity3D**,** peliohjelmointi

ABSTRACT

Oulu University of Applied Sciences
Computer Science, Software Development

Author: Juho Pekkala
Title of thesis: Mobile game development for startup game studio in Unity3D-enviroment
Supervisor: Markku Rahikainen
Term and year when the thesis was submitted: Spring 2013 Pages: 44

Behind the thesis is to create groundings for methods of mobile game development and phased comprehensive plan about the process to create mobile games all the way from scratch to the application store, which are needed in a start up game company.

The goal is to create a research about conditions of mobile game development in Unity3D environment. The research focuses on traditional software development process areas such as suitable software development methodology for the project, project and source code management. At the same time research will also focus on process areas of game development.

Thesis made great groundings for a start up game studio to start mobile game development in Unity3D environment. When the thesis is ready author will be starting part-time entrepreneurship and start developing mobile games and making use of the groundwork the thesis provides.

Keywords: mobile games, mobile game development, startup game company, Unity3D, game development

SISÄLLYS

TERMIEN JA KÄSITTEIDEN SELITYKSET

3D-mallinnus	Kolmiulotteinen mallinnus
3D-objekti	Kolmiulotteinen kappale
Android	Googlen kehittämä mobiilikäyttöjärjestelmä
Boo	Unityssä käytettävä Python-johdannainen ohjelmointikieli
Collision detection	Käsite peliohjelmoinnissa kahden tai useamman objektin kohdatessa tai törmätessä toisiinsa
IDE	Integrated Development Enviroment on ohjelmisto, joka sisältää ohjelmointiin tarvittavan lähdekooditekstieditorin, kääntäjän ja virheiden jäljittäjän
iOS	Applen kehittämä mobiilikäyttöjärjestelmä
Kanban	Tarkoittaa kanban-ideologiaa, joka helpottaa sen suunnittelua, mitä tehdä, milloin tehdä ja kuinka paljon
Kehitysympäristö	Tarkoittaa ohjelmaa tai ohjelmien joukkoa, jolla voidaan toteuttaa ohjelmistoja
Koodauskonventio	Tapa ohjelmoida muiden ohjelmoijien kanssa siten, että yleinen ohjelmointityyli on sama ja näin ollen jokaisen helposti ymmärrettävissä
Linux	Linus Torvaldsin kehittelemä avoimen lähdekoodin käyttöjärjestelmä
Mac OS X	Applen kehittämä käyttöjärjestelmä
Mesh	Kolmiulotteisen objektin verkkomainen runko joka muodostuu kolmikulmioista

Pelimoottori	Pelimoottori on videopelin ohjelmarunko, joka käsittää yleisen pelimekaniikan, kuten esimerkiksi objektien mallintamisen ja piirtämisen, fysiikkamallinnuksen tai tekoälyominaisuuksia
Renderöinti	Renderöinti tarkoittaa kuvan luomista ohjelmallisesta kuvausdatasta
Repo	Repository on versionhallinnassa käytetty termi, joka tarkoittaa palvelimella sijaitsevaa ohjelmistoprojektin lähdekoodin säilytyspaikkaa
Tablet	Kosketusnäytöllinen mobiilitietokone
Tutoriaali	Johdatustyyppinen oppikurssi
Windows	Microsoftin kehittämä käyttöjärjestelmä
XP	Extreme Programming on ketterä ohjelmistokehitysmenetelmä

1 JOHDANTO

Opinnäytetyössä tutustutaan Unity3D-pelinkehityksessä tarvittaviin menetelmiin sekä työkaluihin. Työssä esitellään pienen peliyrityksen tarpeisiin valittuja työkaluja ja kehitysmenetelmiä perusteluineen sekä tutustutaan eri vaiheisiin pelin kehityskaaressa lähtötilanteesta aina sovelluskauppaan asti. Opinnäytetyössä keskitytään esittelemään olennaisimmat asiat Unity3D-kehitykseen liittyen. Työssä ei keskitytä teknisiin toteutuksiin, ratkaisuihin tai ohjelmointiin, sillä siihen materiaalia on saatavilla runsain määrin. Valittujen menetelmien ja työkalujen toimivuutta on testattu käytännössä.

Opinnäytetyön aiheen taustalla on ollut innostus tehdä pelejä ja nykyään olla osa pelialaa ja kehittää ihmisille viihdyttäviä pelejä. Yrityskumppaneinani toimii lisäkseni kaksi muuta opiskelijatoveriani. Pitkien suunnittelujen tuloksena olemme päättäneet lähteä mobiilipelikehityksen linjalle, sillä tällä hetkellä on viihdyttäville mobiilipeleille paljon kysyntää. Päätös lähteä kehittämään mobiilipelejä kesti pitkään ja vaati aika ajoin pohdintoja ja suunnitelmia tulevaisuudelle. Hieman ennen opinnäytetyön aloittamista olimme saaneet Oulun seudun ammattikorkeakoululta mahdollisuuden päästä tekniikan yksikön uusiin yrityshautomotiloihin istumaan rauhassa alas ja palaveeraamaan yrityksen asioista ja saimme asioita vietyä eteenpäin. Siitä lähti kypsymään ideani, että tekisin opinnäytetyönä jotain, joka edesauttaisi yrittämisemme etenemistä.

Syksy 2012 oli ihanteellinen ajankohta aloittaa opinnäytteen työstäminen, sillä suunnitelmissamme olisi yrityskumppaneiden kanssa saada jokainen omansa opinnäytetyön suunnilleen samoihin aikoihin valmiiksi, jotta voisimme aloittaa mobiilipelien kehityksen. Työn aloittaminen alkoi vaivattomasti, sillä olimme jo aikaisemmin yrityskumppaneiden kesken päättäneet kehitysympäristöksi Unity3D:n. Opinnäytetyö etenee läpikäymällä esivalmisteluja ja teoria-asioita, joita on syytä tietää ennen pelin kehittämisen aloittamista.

2 ALUSTUS

2.1 Ohjelmistotuotanto

Ohjelmistotuotannolla tarkoitetaan yleisesti kokonaisuutta, joka sisältää kaikki menetelmät ja vaiheet, jotka kuuluvat työntekoon ja työnjohtoon, kun valmistetaan tietokoneohjelmia tai tietokoneohjelmista koostuvia tietokoneohjelmistoja. Käsitettä ohjelmistotuotanto ei pidä sekoittaa pelkästään ohjelmointiin, vaan laajemmin ohjelmistotuotanto-käsitettä tarkasteltuna se käsittää kaikki mahdolliset vaiheet ohjelman tai ohjelmiston elinkaaressa. (1.)

Ohjelmistotuotannon prosessi voi alkaa esimerkiksi asiakkaan tarpeesta, jonka jälkeen ohjelman tuottaja tekee tarkat määrittelyt asiakkaan tarpeista. Prosessin voi aloittaa myös itse ohjelman tuottaja, mutta se edellyttää selvitystä ohjelmiston tarpeellisuudesta markkinatutkimuksen perusteella. Edellä mainittujen vaiheiden jälkeen ohjelmistotuotannossa voidaan siirtyä ohjelmistokehityksen prosesseihin, joita ovat vaatimusten ja toiminnallisuuksien määrittely, arkkitehtuurin ja ulkoasun suunnittelu, toteutus, ohjelmistotestaus, käytettävyystestaus sekä käyttöönotto ja ylläpito. (1.)

2.1.1 Ohjelmistokehitysmenelmät

Ohjelmistokehitysmenetelmän valinta ohjelmistojen kehittämiselle on erittäin tärkeä niin ohjelmistoprojektin onnistumisen kuin yrityksenkin kannalta. Oikean ohjelmistokehitysmenetelmän valitseminen ja käyttäminen edesauttaa hallitsemaan ohjelmistoprojekteja systemaattisesti. Menetelmillä pyritään mallintamaan ohjelmistojen valmistusprosessia koko kehityksen elinkaaren mukaisesti. Ohjelmistokehitysmenetelmiä on monia erilaisia, joten yrityksillä menetelmien valintaan vaikuttavat paljon yrityksen koko sekä projektien luonne ja laajuudet. (2.)

Eroavaisuudet menetelmien välillä tulevat lähinnä lähestymistavoista projekteihin ja siitä, millainen on kehitettävä ohjelmisto. Esimerkiksi pienessä yrityksessä voidaan ottaa käyttöön menetelmä, jossa ei käytetä paljoa aikaa projektin tarkkaan dokumentointiin ja projektia edeltävään yltiöpäisen kattavaan

suunnitteluun vähäisen työntekijämäärän vuoksi. Isommissa yrityksissä on enemmän resursseja käytettävissä, jolloin projektien elinkaaren vaiheet voidaan dokumentoida huolellisesti. Kattava sekä tarkka dokumentointi suuremmissa yrityksissä ovat elintärkeitä keinoja saada pidettyä suuri määrä työntekijöitä ajantasalla projektien vaiheista. (2.)

Olen päätynyt käyttämään ja soveltamaan pelinkehityksessä ketteriä menetelmiä. Erityisesti olen tutustunut eXtreme Programmingiin, Kanbaniin sekä sekä Scrumiin. Kaikista kolmesta olen kerännyt itselleni ja pienen kolmihenkisen yritykseni tarpeisiin toimivia ja sopivia työskentely- ja dokumentointitapoja.

XP:n eli eXtreme Programmingin pääperiaatteet ovat kehittyneet ohjelmistokehityksen parhaimmista menetelmistä. XP on kevyt, tehokas, joustava tapa kehittää ohjelmistoja. Menetelmässä avainasemassa ovat minimaalinen dokumentointi, pariohjelmointi, testivetoinen kehitys, jatkuva integraatio, refaktorointi, koodauskonventiot, koodin yhteisomistajuus ja Keep It Simple and Short -periaate. XP on keveytensä puolesta ihanteellinen menetelmä pienelle kehitystiimille, sillä XP:ssä keskitytään olennaiseen eli itse tekemiseen. Lyhyin väliajoin pystytään saamaan näkyviä tuloksia aikaan, mikä on pienelle tiimille myös olennainen motivaation lähde. XP:n metodien avulla projektin riski on erittäin pieni sekä projektin suuntaa voidaan muuttaa esimerkiksi keskellä projektia. (3, s. xv–xvii.)

Scrumin pääperiaatteet ovat holistinen lähestymistapa, projektin vaiheistaminen sekä jatkuva projektin etenemisen kontrollointi. Scrumin mukaan yksi monitaitoinen tiimi suorittaa kehitysprosessin alusta loppuun asti vaiheistuksella, joka on vahvasti jaoteltu. Menetelmän ideana on pyrkiä etenemään projektissa yksikkönä ja toimimaan tiiviissä yhteistyössä. (4.)

Omiin tarpeisiin omitut työskentelytavat kallistuvat pitkälti XP:n suuntaan pois lukien parikoodaaminen ja suunnittelupuolelta olen valinnut Scrumista vaiheistetun suunnittelun. Vaiheistetulla suunnitelulla tarkoitan sellaista suunnittelua, jossa otetaan esille iso määrä ideoita, joiden tärkeys priorisoidaan. Näin ollen voidaan priorisoida tärkeät ominaisuudet, jolloin oleelliset asiat

toteutetaan ensimmäisessä vaiheessa ja siirretään vähemmän oleellisia ominaisuuksia ja toiminnallisuuksia seuraaviin vaiheisiin. Tämän avulla voidaan suunnitella ja hahmotella projektin kokonaiskuvaa yleisellä tasolla, sekä sen pohjalta on helpompi asettaa tavoitteita sekä aikataulua projektin suhteen.

2.1.2 Projektin- ja versionhallinta

Projektin- ja versionhallinnalla tarkoitetaan tapaa hallita ohjelmistoprojektin tuotoksia ja seurata kehityksen etenemistä ja näin ollen helpottaa hallittua ohjelmistokehitystä.

Pienessä kehitystiimissä projektinhallinta on tiimin omalla vastuulla. Pienessä tiimissä voi olla määritelty jollekin jäsenelle projektipäällikön rooli, jonka vastuulla on projektin kokonaishallinta. Pienessä kehitystiimissä tämä ei myöskään ole välttämätön. Pienessä kehitystiimissä jäsenten on helppo keskenään sopia projektiin liittyvät tehtävät esimerkiksi osaamisen tai halukkuuden pohjalta. Lisäksi on olemassa useita erilaisia tehtävien hallintaohjelmistoja, kuten esimerkiksi Trello, joka esitellään myöhemmin. Tehtävänhallinnan avulla voidaan jäsenten kesken hallita projektia jakamalla tai valitsemalla itse tehtäviä, jotka merkitään tehdyksi, kun ne ovat valmiita. (5.)

2.2 Työvälineet sekä hankintojen kartoitus

Aloittavalle peliyritykselle on enemmän kuin tarpeen kartoittaa alkuun tarvittavat työvälineet ja niiden kustannukset. On hyvä selvittää, millaiset tietokoneet ja ohjelmistot ovat tarpeen. Oikeiden työkalujen ja ohjelmistojen kartoitus on tärkeää. Oikeanlaisilla ohjelmistoilla saadaan kehitysprosessista selkeä, saadaan kehittämisestä mielekästä, sekä onnistuminen on askeleen lähempänä kun valitaan alusta asti oikeat ohjelmistot. Yrittäjille on hyvä saada myös kokonaiskuva siitä, millaisia ohjelmistoja alkuun pääsemiseksi tarvitaan.

Vertailun vuoksi on valittu ohjelmistoista, tietokoneista ja testilaitteista esimerkkejä kaksin kappalein. Hankintojen alkuun on myös hyvä tuoda esille itseltään kysyttävät kysymykset: mikä on haluttu mobiilipelinkehityksen kohdealusta sekä onko peli suunnattu älypuhelimelle vai tablet-tietokoneelle, sillä ne vaikuttavat valittaviin työvälineisiin. Mobiilipelinkehitystä ajatellen

vaihtoehtoja ovat tällä hetkellä Unity3D-ympäristössä Android ja iOS (6). Unity3D:lle on tulossa mahdollisuus myös kehittää Windows Phone 8 -alustalle (7).

2.2.1 Tietokoneet sekä testilaitteet

Valittavista tietokoneista löytyy erittäin paljon valinnan varaa. Työkoneisiin otetaan tarkasteluun vain ne tietokoneet, joissa on käytössä käyttöjärjestelmänä Microsoft Windowsista tai Mac OS X:stä. Perusteluni näille kahdelle vaihtoehdolle on pelimoottorin ja kehitysympäristön tuen puuttuminen Linux-pohjaisilta käyttöjärjestelmiltä opinnäytetyön kirjoitushetkellä. Vaikka Windows-pohjaisille tietokoneille voikin asentaa pienellä vaivalla Linux-käyttöjärjestelmän, niin se ei välttämättä takaa Linuxin täydellistä käyttöä. Hankaluuksia voi esiintyä erityisesti tietokoneen eri komponenttien Linux-ajurien puutteen vuoksi.

Microsoft Windows -käyttöjärjestelmällä varustettuja valmiita pakettitietokoneita on tarjolla lukuisilta eri valmistajilta edullisesti. Windows-PC:n voi myös vaivattomasti koota myös valitsemistaan komponenteista. Näin ollen saa helposti räätälöityä tarpeisiin sopivan kokoonpanon, joka mahdollistaa myös jatkossa komponenttien päivittämisen vaivattomasti. Mobiilipelinkehityksen aloittaminen Unity3D-ympäristössä Windows-PC:llä mahdollistaa ainoastaan Android-alustalle kehittämisen ja myöhemmin tulevalle Windows Phone 8:lle.

Applen Mac-tietokone on pakollinen hankinta, mikäli on päättänyt aloittaa kehittämisen iOS-alustalle, koska vain Macillä pystytään ohjelmoimaan ja kääntämään ohjelmia Applen käyttöjärjestelmille. Mac-tietokoneiden huono puoli on laitteiden korkeahko hinta.

Tuotettujen pelien laadun varmistaakseen on syytä myös hankkia asiaan kuuluvia testilaitteita. Testilaitteita hankkiessa kannattaa ottaa selvää sen hetken markkinoiden suosituimmista malleista ja perustaa hankintapäätös sen mukaisesti. Mikäli on varaa sijoittaa useampiin testilaitteisiin, silloin on hyvä hankkia eri valmistajien malleja sekä uudempaa että vanhempaa mallia.

2.2.2 Palvelimet

Aloittavan pienen peliyrityksen alkutaipaleella ei välttämättä ole tarpeellista hankkia erillisiä palvelimia kustannusten vuoksi, ellei alusta lähtien ole tarkoituksena luoda peliä moninpelio minaisuuksilla varustettuna. On syytä ottaa huomioon, että moninpelio minaisuuksilla varustetun pelin ajaminen palvelimella voi vaatia suurta sijoitusta rahallisesti mikäli sen täytyy pystyä kestämään suuret pelaajamäärät ja niiden moninpelit.

Yrityksen alkutaipaleella ei siis välttämättä tarvita ulkopuolisia palvelimia, sillä valmiita pelejä voidaan myös pitää sovelluskauppojen palvelimella. Julkaisukelpoiset pelit lähetetään sovelluskauppojen haltijoille tarkistettavaksi ja kun peli on hyväksytty, se otetaan sovelluskaupan haltijan palvelimelle, josta loppukäyttäjät eli asiakkaat voivat pelin ostaa ja ladata. Sovelluskauppojen haltijat kuten esimerkiksi Apple tarjoavat myös mahdollisuuden moninpelio minaisuuksiin, kuten pelaajan piste- ja saavutustietojen keräämisen palvelimelle (8.). Sovelluskauppojen haltijoiden tarjoamat palvelinpalvelut ovat kuitenkin rajoittuneet, joten pelinkehittäjien on syytä itse kartoittaa omat tarpeet ja sen mukaan ottaa tarkemmin selvää millaisia resursseja palvelimelta vaaditaan.

2.2.3 Pelimoottori ja kehitysympäristö

Pelimoottoriksi ja kehitysalustaksi on valittu Unity3D. Unity3D toimii koko käsiteltävän aihepiirin pohjana ja ytimenä. Unity3D-kehitysympäristössä tulee mukana .NET-pohjainen IDE nimeltä MonoDevelop. MonoDevelop on avointa lähdekoodia ja siitä on versiot Linux-, Max OS X- ja Windows-ympäristöille. Unity3D:llä kehitettäessä on myös mahdollisuus käyttää muitakin kehitysympäristöjä ja editoreita, jotka soveltuvat Unityn tukemien kielien JavaScriptin, C#:n ja Boon kirjoittamiseen. Kolmannen osapuolen editoreita löytyy useita, joten niiden käyttäminen on käyttäjistää itsestään kiinni ja pitkälti makuasia. Lisäksi kolmannen osapuolen editoreiden kaveriksi tarvitaan tietysti kääntäjä. Näin ollen eri mahdollisuuksia syntyy liikaa, joten vertailusta tulisi liian valtaisa. Unityn mukana tuleva MonoDevelop on jo itsessään mainio

kehitysympäristö ja se sisältää oman kääntäjän, joten on suositeltavaa aloittaa käyttäen MonoDevelopia.

Unity3D on täysin integroituva pelimoottori ja täysi työkalusetti, joka tarjoaa intuitiivisen kehitysympäristön sekä nopean ja tuotteliaan työnkulun, jonka ansiosta kehittäjät voivat luoda pelejä huomattavasti nopeammin, pienemmällä vaivalla ja rahalla. Unity on suunniteltu pitkälti käyttäjälähtöiseksi, joten se tarjoaakin laajat ja sukkelat toiminnallisuudet luoda pelejä ja interaktiivista 3D-sisältöä. Vaikka pelimoottori on nimeltään Unity3D, voidaan sillä huoletta myös luoda 2D-pelejä. Esimerkiksi Rovion tekemä Bad Piggies on tehty Unityä käyttäen (9). Unityllä voidaan luoda ja julkaista pelejä kymmenelle eri alustalle kuten pöytätietokoneiden alustoille, flash-pohjaisena webbiin, iOS:lle, Androidille, Wiille, PS3:lle sekä XBOX 360:lle. (6.)

2.2.4 3D-mallinnus- ja animointiohjelma

3D-mallinnus- ja animointiohjelma on elintärkeä pelinkehityksessä. Sen avulla saadaan luotua peliin objekteja joista pelimaailma pitkälti koostuu. Animoinnilla saadaan luodut 3D-objektit liikkumaan ja niin sanotusti elämään. Erillisiä 3D-mallinnusohjelmia on monia mistä valita ja Unity tukee esimerkiksi seuraavia 3D-mallinnusohjelmia: Maya, 3ds Max, Modo Cinema 4D, Cheetah 3D ja Blender (10). Vertailun vuoksi on esiteltäväksi valittu kaksi ohjelmaa, joista toinen on ilmainen ja toinen maksullinen ohjelma.

Autodesk tarjoaa ammattilaiskäyttöön suunnatun 3ds Max-3D-mallinnus- ja animointiohjelman, joka on suunnattu pelinkehittäjille, visuaalisten efektien ja liikkuvan grafiikan taiteilijoille sekä luoville ammattilaisille media-alalla. Lisenssi 3ds Maxiin on suhteellisen kallis ja iso sijoitus aloittavalle peliyritykselle. (11.)

Blender on erittäin suosittu ja ilmainen 3D-grafiikan mallinnusohjelma, jota levitetään GNU GPL-lisenssin alaisena (12). Blenderin valinta 3D-mallinnukseen oli helppo, sillä sen käyttämisestä on kokemusta jo aiemmin suoritetulta pelinohjelmointikurssilta ja toisena tärkeänä perusteena on se, että Blender on ilmainen.

2.2.5 Kuvankäsittelyohjelma

Kuvankäsittelyohjelma on myös tarpeen pelinkehitykseen. Pelien grafiikat luodaan pääasiallisesti kuvankäsittelyohjelmalla. 3D-mallinusohjelmalla tehdään runko ja kuvankäsittelyohjelmalla saadaan 3D-mallinnukset niin sanotusti puettua. Vertailuun on jälleen valittu kaksi ohjelmaa, joista toinen on ilmainen ja toinen maksullinen ohjelma.

GIMP on ilmainen avoimen lähdekoodin kuvankäsittelyohjelma, sekä se toimii useimmilla käyttöjärjestelmillä. GIMP tarjoaa suhteellisen hyvät työkalut grafiikan luomiseen. (13.)

Adobe Photoshop on erittäin ollut suosittu ja tehokas grafiikan luomiseen ja muokkaamiseen tarkoitettu ohjelma. Photoshop tunnetaan niin sanotusti ammattilaisten ohjelmana. Photoshopissa on erittäin suuri valikoima työkaluja ja näin ollen se on oiva valinta kuvankäsittelyohjelmaksi. Itse valitsin Adobe Photoshopin projektiini grafiikkatyökaluksi, sillä omistan ennestään Adobe Photoshop CS5:n. Mainitsemisen arvoinen on myös Adoben tarjoamat huikeat opiskelijaetuudet Adoben tuotteisiin, joka koskee myös Photoshopia (14).

2.2.6 Versionhallinta

Versionhallinta on tärkeä osa projektien hallinnassa varsinkin jos kehitystiimiin kuuluu useita henkilöitä. Versionhallintaan löytyy valinnanvaraa runsaasti, joten esittelen muutaman yleisimmin käytetyt monen käyttäjän versionhallinnat. Unity-pelinkehityksessä on varaa valita Unityn oman Team Licensen tai perinteisemmän versionhallinnan väliltä. Unityn oma Team License on helppo ja yksinkertainen valinta sellaiselle, joka ei halua konfiguroida erillistä versionhallintaa. Unityn Team License maksaa suhteellisen paljon verrattuna vuokrapalvelimiin tai GitHubiin. Kannattaa myös ottaa huomioon, että nimi Unity Team License on hieman harhaanjohtaja. Jokaisella kehittäjällä on oltava oma Team License ja se vaatii myös oman palvelimen asennuksineen siitä huolimatta. (15.)

Versionhallintaan tarvitaan ensinnäkin palvelin, jossa projekteja ja niiden versioita voidaan hallita. Aloittavalle peliyritykselle suosittelen lämpimästi

avoimen lähdekoodin versionhallintatyökaluja, sillä ne tulevat huomattavasti edullisemmaksi verrattuna kaupallisiin systeemeihin. Edullisesta tietokoneesta voi helposti saada palvelimen yrityksen sisäiseen käyttöön, kuten esimerkiksi versionhallintaan ym. yrityksen sisäisiin asioihin. Hyviä vaihtoehtoja versionhallintaohjelmistoista olisivat esimerkiksi Subversion sekä Mercurial. Monilla vuokrapalvelimien tarjoajilla on tarjota Subversion helposti asennettavana moduulina.

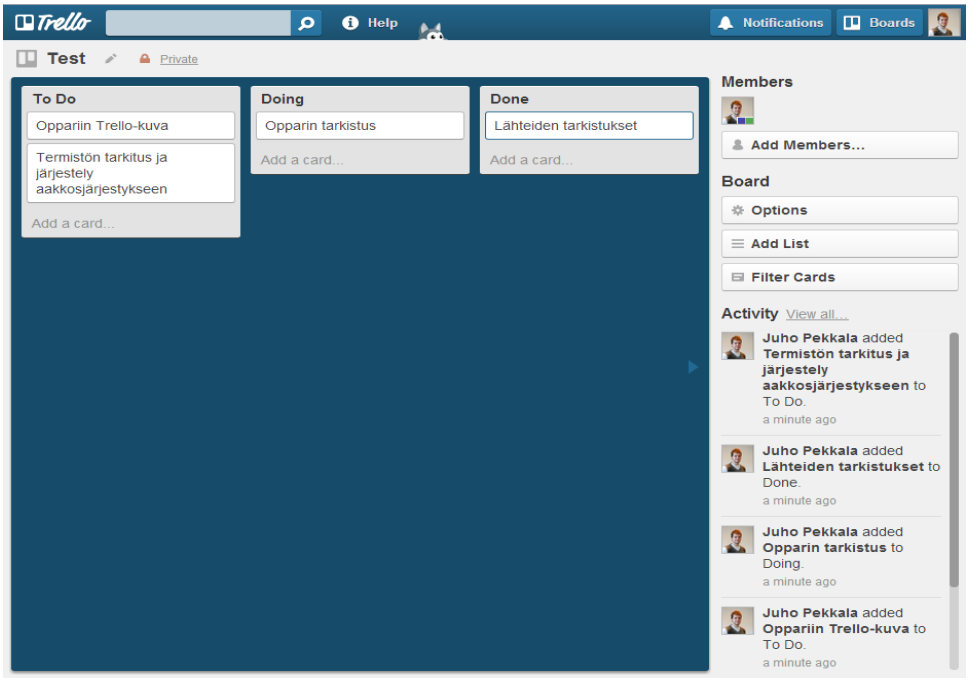
Mikäli palvelimen asentaminen ja vuokraaminen ei miellytä, on myös olemassa versionhallintapalveluita, kuten esimerkiksi GitHub. GitHub ei ole pelkästään versionhallintaohjelmisto vaan myös palvelu, joka tarjoaa projektien versionhallinnan hostingin. Githubin parhaat puolet ovat sen lisäksi, että se on git-pohjainen, myös se, että GitHub tarjoaa web-pohjaisen graafisen käyttöliittymän versionhallinnan seurantaan. Lisäksi web-käyttöliittymässä on mahdollisuus rajata pääsyoikeuksia tiedostoihin ja monia yhteistyöhön tarvittavia toimintoja, kuten wikejä ja tehtävienhallintatyökalut jokaiseen projektiin. GitHub on ilmainen, mutta mikäli haluaa yksityiset repot eli lähdekoodien säilytyspaikat on palvelusta maksettava vähintään 7 dollaria kuukaudelta. (16.) Mielestäni hinta on erittäin edullinen ja tarjoaa rahaa vasten erittäin hyvän palvelun. Lisäksi pientä maksua vasten välttyy siltä, että ei tarvitse konfiguroida itse versionhallintaa omalle tai vuokrapalvelimelle.

2.2.7 Projektin- ja tehtävienhallinta

Versionhallinnan ohella on hyvä olla käytössä projektin- ja tehtävienhallintaohjelma. Sen avulla voidaan helposti pilkkoa projektia pieniksi tehtäviksi ja jakaa tehtäviä projektiin osallistuvien henkilöiden kesken. Näin saadaan projekti etenemään nopeammin ja sujuvammin.

Projektin- ja tehtävienhallintaohjelmia on suunnaton määrä, joten niistä on lähes mahdoton luoda järkevää vertailua. Aloittavalle yritykselle suosittelisin kuitenkin web-pohjaista avoimen lähdekoodin ohjelmaa. Esimerkiksi Trello on erittäin hyvä ohjelmisto. Trellon projektinhallinta perustuu kanban-ajatusmalliin. Trello on konkreettisesti taulu johon voidaa luoda listoja, jotka toimivat tehtävälistoina. Listojen sisälle luodaan kortteja, jotka toimivat tehtävinä. Korttien on tarkoitus

on liikkua listoista toiseen (kuva 1) vuon tavalla. Esimerkiksi tehtävä etenee idealistasta toteutukseen. Lisäksi Trellolla on myös mobiiliversio, joka on erittäin hyvä ideoiden kirjailuun sekä tehtävien lisäämiseen myös kun ei ole työpisteen äärellä. (17.)



KUVA 1. Trellon kanban-taulu

2.2.8 Lisenssit

Lisenssien hankkiminen on välttämätöntä. Riippuen kehiteltävien pelien kohdealustasta on hankintalistalle laitettava Unity3D:n iOS-lisenssi mikäli aikoo kehittää Applen mobiililaitteille pelejä, ja vastaavasti Unity3D:n Android-lisenssi. Lisäksi välttämätön hankinta on myös Applen tai Googlen vuosittain maksettava kehittäjälisenssi. Näiden lisäksi lisenssit on oltava kunnossa pelinkehitykseen tarvittavissa ohjelmistoissa, jotka ovat kaupallisia. Unity3D:stä on olemassa myös Pro-tason lisenssi, joka avaa lisäominaisuuksia pelinkehittämiselle. Kääntöpuolena Pro-tason lisenssissä on se, että se tarkoitettu yrityksille joilla liikevaihto on edellisvuotena ylittänyt 100 000 dollaria. Mikäli liikevaihto ylittää satatuhatta dollaria, on yrityksen ostettava Unity Pro –lisenssi. (18.)

2.2.9 Sovelluskaupat

Sovelluskaupat ovat paikkoja joihin voi saada omia pelejä myyntiin. Niitä ovat Applen App Store ja Androidille Google Play. Sovelluskaupat on oikeastaan paras keino ja kanava saada pienen yrityksen pelejä tarjolle ja myytyä. Sovelluskaupat ottavat kuitenkin 30 prosenttia myydyn sovelluksen hinnasta, joten kehittäjälle jää 70 prosenttia. (19; 20.)

2.2.10 Vertailutaulukot työkalujen hankinnoista

Hankintataulukon on tarkoitus olla suuntaa-antava, sekä selkeyttää lukijan näkemystä siitä mihin varoja voi kulua. Hankinnat on laskettu henkilöä kohti -periaatteella. Laitevalinnat ovat täysin kirjoittajan omakohtaisia valintoja ja valitut laitteet eivät ole välttämättömiä kehityksen kannalta, sillä on olemassa vaihtoehtoisiakin laitteita ja vanhemmat laitteet voivat myös hyvin toimia testilaitteistona. Optimoinnin kannalta on syytä kuitenki ottaa huomioon, että hankkii testilaitteen kohdealustan mukaan. Tietokonepaketteihin on sisällytetty oleelliset oheislaitteet, kuten näyttö, näppäimistö ja hiiri.

Hinnat ovat poimittu verkkokaupoista sekä ohjelmien kotisivuilta 2013 alkuvuodesta. Hinnat ovat suuntaa antavia ja voivat vaihdella ajankohdasta ja paikan mukaan.

TAULUKKO 1. Hankintojen kartoitus

Työväline	Alkuun pääseminen Mac & iOS	Alkuun pääseminen Windows PC & Android	iOS & Android ja kaupalliset lisenssit	Kirjoittajan valinnat	Hinta (€, sis. alv)
Mac mini-paketti	X		X	X	1000
PC-paketti		X	X		1000
iPhone 5 16gb	X		X	X	689
iPad 16gb wifi	X		X	X	500
Asus Padfone		X	X		500
Blender 3D	X			X	Ilmainen
Autodesk 3ds Max			X		2720
GIMP	X	X			Ilmainen
Photoshop			X	X	920 / opiskelijalle 230
Apple Developer License	X		X	X	74
Android Developer License		X	X		19
Unity iOS License	X		X	X	381
Unity Android License		X	X		381
Unity Team License			X		475
GitHub	X	X	X	X	63 / vuosi
Yhteensä (€)	2707	1963	8722	2937	

3 PELISUUNNITELMA

Pelinkehityksen kaaren alkuvaiheessa on suositeltavaa laatia alustava pelisuunnitelma. Pelisuunnitelmasta kannattaa tehdä kattava. Kattava suunnitelma helpottaa hahmottamaan tulevan pelin kokonaisuutta. Lyhyenä nyrkkisääntönä on se, että pelissä on aina alku ja loppu, ja kaikki muu mitä peliin suunnitellaan sijoittuu johonkin niiden väliin. (21.)

3.1 Resurssit

Yleisesti ennen pelisuunnitelmaa ja peli-idean suunnittelua on hyvä ottaa selvää olemassa olevista resursseista ja siitä, mistä niitä tarvittaessa olisi saatavilla. Alkuun on syytä selvittää pelintekijöiden kyvyt ja rajat: mitä he osaavatt, mitä he eivät osaa ja onko mahdollista tai tarpeellista opetella uusia asioita.

Suunnitelman edetessä kykyjen ja rajojen mukaan kannattaa myös jakaa projektissa tehtävät ja myös mahdollisesti rooleja, ettei tehtävien tekemisessä synny epäselvyyksiä ja erimielisyyksiä. Matemaattisesti lahjakkaan kannattaa suunnitella pelin logiikkaa ja monimutkaiseen matematiikkaan liittyviä asioita, joita peruspelaaja ei koskaan tule pelatessaan ajatelleeksi. Taitavan piirtäjän kannattaa piirtää runsaasti konseptitaidetta innostamaan muita projektin jäseniä ideoimaan pelimekaaniikkaa ja sisältöä. Mikäli peliin tarvitaan musiikkia, mutta omasta porukasta ei muusikon alkua löydy niin kannattaa muistaa, että esimerkiksi tunnusmusiikit voi saada musiikkiopistolta tai alan opiskelijoilta edullisesti. (22, s. 37.)

Pelinkehityksessä on olemassa tietysti osa-alueita, joita ei kannata lähteä opettelemaan tai edes yrittämään projektia aloittaessa. Jos ei osaa ohjelmoida, ensimmäisenä projektina pelin ohjelmoiminen ei ole ehkä parhain tapa oppia sitä. Jos ei osaa piirtää, ei tulevan pelimaailman konseptien loihtiminen ole kovin järkevää. Vaikka pelien tekeminen opettaa tekemään pelejä entistä paremmin, on kouluttautuminen syytä olla hoidettuna etukäteen. (22, s. 37.)

3.2 Pelin kuvaus

Pelin kuvauksella tarkoitetaan lyhyttä kuvausta pelistä. Pelin kuvaus on kuin kirjan takakansi, joka kertoo oleelliset asiat tiivistettynä. Pelin kuvaus voi kertoa millainen peli on, mihin genreen peli sijoittuu, mikä on pelin slogan, millainen tarina lyhyesti on ja millainen on pelaajan rooli pelissä. (21.)

3.3 Pelitoteutuksen perustelu

Ennen peliprojektin aloittamista on syytä perustella miksi peli ansaitsee tulla toteutetuksi. Kannattaa pohtia onko peli-ideassa jotain ennen näkemätöntä tai onko jokin idea jo olemasa, jota parannellaan uusilla ideoilla. (21.)

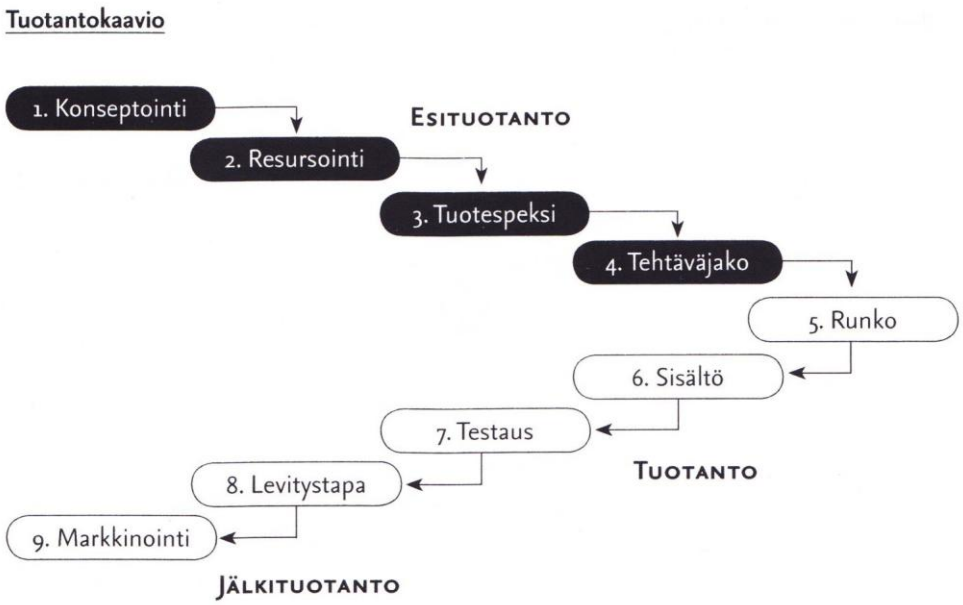
3.4 Kohderyhmä

Peliä suunniteltaessa on tärkeä ymmärtää kenelle peliä tehdään. Useasti pelejä suunniteltaessa unohtuu itse käyttäjä, ihminen joka peliä tulee pelaamaan. Pelin kehittäminen voi joutua sivuraiteille, jos unohdetaan alkuperäinen kohderyhmä ja ne ihmiset joille peli itseasiassa tehdään. Se voi helposti aiheuttaa sen, että peli sisältää suuret määrät ylimääräisyyksiä, jolloin yhtenäinen kokonaisuus ja pelattavuus on kärsinyt. Sen vuoksi on ajateltava alusta alkaen kohderyhmää eli ihmisiä, jotka pelin mahdollisesti ostavat ja sen pelaavat. Pelistä ei ole tarkoitus vain saada pelkkää tuotetta vaan myös luoda tuote jota ihmiset haluavat mielellään pelata. Pelin on tarkoitus tuoda käyttäjälleen kokemus tai elämys (23, s. 10). On myös otettava huomioon, että on kohderyhmä mikä tahansa, aikuiset, nuoriso, lapset, miehet tai naiset, tytöt tai pojat, se voi vaikuttaa huomattavasti myös pelin sisällön suunnitteluun. Kohderyhmän myötä voidaan myös pohtia pelin genreä ja sen tyypillisiä pelaajia. Toisaalta pelisuunnittelussa voi käydä myös niin, että ensin toteutetaan visio pelistä ja sen jälkeen pohditaan millaiselle kohderyhmälle peli sopii.

3.5 Tuotantosuunnitelma

Tuotantosuunnitelmat suurissa peliyhtiöissä ovat arkipäivää ja niitä voivat hyödyntää pienetkin peliyritykset. Yleensä tuotantosuunnitelman avulla voi säästää paljon aikaa ja vaivaa kun pyrkii pitäytymään alkuperäisissä

suunnitelmissa ja työnvaiheissa. Tuotantosuunnitelmalla on tarkoitus selventää pelin toteuttamisprosessia ja työn vaiheita (kuva 2). Tuotantosuunnitelmaan selvitetään muun muassa menetelmät, työkalut, prosessit sekä kohdealusta. Lisäksi on myös hyvä ottaa huomioon olemassa olevat resurssit, jotka helpottavat kartoittamaan tuotannon laajuutta hieman tarkemmin. (21; 22, s. 40.)



KUVA 2. Tuotantokaavio (22, s. 41)

3.6 Avainideat

Pelisuunnittelussa tärkeässä asemassa on tulevan pelin avainideoiden selvittäminen. Esimerkiksi moninpeleissä päällimmäisenä avainideana on pelaaminen yhdessä muiden pelaajien kanssa. Tavoitteena on tehdä yhteistyötä tai luoda tiimi, jolla pyritään yhdessä saavuttamaan pelissä jokin tavoite tai voittamaan kilpaileva tiimi. Avainideoita suunniteltaesa on syytä miettiä, mitä valitut ideat tuovat itse peliin ja miten valitut ideat myyvät peliä. (21.)

3.7 Taustatarina

Peliin on tärkeää suunnitella taustatarina. Taustatarinalla saadaan luotua peliin ainutlaatuista sisältöä joihin pelaaja voi samaistua ja uppoutua. Taustatarinaa

suunniteltaessa on hyvä esimerkiksi miettiä pelin aloitustilannetta ja sitä, mitä on tapahtunut ennen aloitustilannetta ja miten tarina etenee aloitustilanteesta. Taustatarinaan kannattaa esimerkiksi pohtia mikä on itse pelaajan rooli tarinassa, missä pelaaja on, mistä pelaaja aloittaa ja mitä pelaajan täytyy tehdä. Lisäksi taustatarina voi toimia myös käsikirjoituksena, jossa voi olla esimerkiksi juonenkäänteitä. (21.)

3.8 Pelin aloitus

Pelin aloituksen miettiminen on tärkeä osa pelissä, sillä se antaa pelaajalle ensivaikutelman pelistä. Pelin aloitus voidaan miettiä periaatteessa taustatarinan mihin tahansa kohtaan, mutta on syytä pohtia millaisen alkutilanteen haluaa pelaajalle antaa, millaiseen tilanteeseen pelaaja asetetaan ja miten se vaikuttaa pelaajan ensivaikutelmaan pelistä ja miten pelaaja tulee näkemään pelin alun. Lisäksi pelin aloitukseen olisi hyvä pohtia millaisia valintoja pelaaja voi tehdä pelin alussa. Pelin alussa voi olla valittavana kenties tietty hahmoluokka tai peli alkaa introlla, joka luo alkutunnelmaa ja esittelee taustatarinan vetovoimaa. (21.)

3.9 Pelimaailma ja kenttäsuunnittelu

Pelimaailman ja kenttien suunnittelussa pureudutaan tarkemmin siihen minne peli sijoittuu. Mikäli pelillä on kattava tarina tai käsikirjoitus olemassa niin siihen on jo silloin vastattu. Taustatarinan pohjalta on hyvä lähteä suunnittelemaan alustavaa pelimaailmaa ja kenttiä. Esimerkiksi Valven suositun Counter-Strike -pelin taustatarinana on terroristien ja anti-terroristien välinen taistelu erilaisissa ympäristöissä ja kamppailu tiimien välisestä voitosta. Pelissä pelimaailma ja -kentät ovat suunniteltu toimiviksi useiden pelaajien muodostamien tiimien väliseen kamppailuun.

Suunnittelussa kannattaa myös ottaa huomioon minkä tyylinen peli on suunnitelmissa. Esimerkiksi seikkailupeli voi olla pitkälti tarinavetoinen ja teknisesti pelimaailmaltaan putkijuoksua. Haastavimpana pelimaailmasuunnittelussa ovat MMO-tyyppiset pelit eli massiiviset moninpelattavat verkkopelit joissa on tietty alue, jossa pelaaja voi täysin

vapaasti kulkea pelikentän reunojen sisällä. Itse pelikenttäsuunnitteluun on olemassa hyviä apuvälineitä ja erilaisia ohjelmia ja map editoreita joilla voidaan suunnitella kenttien pohjapiirustuksia. Hyvä ja ilmainen avoimen lähdekoodin esimerkki on TileStudio.

3.10 Pelattavuus

Pelimaailman ja kenttäsuunnittelun lisäksi on otettava huomioon pelin pelattavuus. Pelattavuus on subjektiivinen käsite, mutta esimerkiksi se voi tarkoittaa sitä, mitä pelaaja kokee peliä pelatessaan ja miten pelaaja itse voi vaikuttaa pelin kulkuun ja liikkumiseen pelimaailmassa. On tärkeää ottaa huomioon pelattavuuden miettimisessä se, että minkä tyyppiset ovat pelin kentät. Pelin kenttätyypit voi jakaa helposti seuraaviin kategorioihin: putkijuoksu, monireitti ja vapaajuoksu. Jos esimerkiksi peli on suunniteltu tarinapainoitteiseksi, niin peli on yleensä putkijuoksumainen eikä peliin tarvitse välttämättä panostaa tekemällä suuria määriä oheisosia tai yksityiskohtia, jonka vuoksi pelaaja ajautuu turhaan tarinan raiteilta. (21.)

Pelattavuus voi käsittää myös pelin kontrollit eli toimintapainikkeet. On tärkeää, että pelin kontrollien käyttäminen on miellyttävää. Kehnosti toteutetut kontrollit voivat ajaa pelaajan pois ennen kuin pelaaja on edes päässyt peliin sisälle. Esimerkiksi mobiililaitteilla on hyvin tärkeä tehdä hyvin toimivat ja selkeät kontrollit, sillä kyseessä ovat kosketusnäytölliset laitteet. Laitteiden ruudut ovat pienet ja näin ollen on otettava huomioon vaikkapa kontrollien sijainnit ruudulla ja riittävä koko, että myös isosorminen pelaaja osuu painikkeeseen. (21.)

3.11 Käyttöliittymä

Jokaisessa videopelissä täytyy olla käyttöliittymä. Erilaisia käyttöliittymiä on yhtä paljon kuin on käyttöliittymäsuunnittelijoitakin, mutta pelien käyttöliittymäkonseptit rajoittuvat muutamiin perspektiiveihin kuten esimerkiksi kaksi- tai kolmiulotteisiin tai ensimmäisen tai kolmannen persoonan kuvakulmiin.

Käyttöliittymää suunnitellessa on hyvä pohtia millaisia toimintoja, painikkeita ja valikoita tarvitaan käyttöliittymään. Kun käyttöliittymään tarvittavat asiat on

kartoitettu, on niistä hyvä tehdä erilaisia vedoksia. Selkeä ja yksinkertainen käyttöliittymä ei voi mennä ikinä pieleen. Hyvänä muistisääntönä voidaan pitää, ettei käyttöliittymässä ole yhtään niin sanottua turhaa elementtiä. Graaffisesti käyttöliittymä pitää totta kai saada sopimaan pelin teemaan sopivaksi. Hyvä on muistaa, että yksinkertainen on aina kaunista joten sanonta ”Keep It Simple and Short” on hyvä laittaa korvan taakse. (21.)

3.12 Tavoitteet

Pelissä kuin pelissä on olemassa tavoitteet. Pelin tavoitteita on hyvä pohtia jo pelinsuunnittelun varhaisessa vaiheessa. Pelitavoite voi olla vaikkapa yksi pelin koukuista. Esimerkiksi pelissä voi hahmolle kerätä kokemuspisteitä ja tarpeeksi kokemuspisteitä keränneenä palkitaan pelaaja hahmo tason nousemisella, mikä taas toisi pelaajalle lisäpalkintona esimerkiksi päivityksiä hahmon varusteisiin tai vaikkapa mahdollisuuden päästä uudelle kentälle tai seuraavalle tasolle. Esimerkiksi yleisenä trendinä mobiilipeleissä on luoda pieniä kokonaisuuksia kenttien muodossa joita läpäisemällä saadaan suorituspisteet ja esimerkiksi arvosana tähtinä ja päästään aina seuraavaan kenttään ja lopulta pelin läpi pääsemiseen. (21.)

3.13 Ominaisuudet

Peliä suunniteltaessa on hyvä pohtia samalla peliin tulevia ominaisuuksia. Pelin ominaisuudet vaikuttavat suuressa määrin pelattavuuteen. Esimerkiksi avaruusaluksella voi olla ominaisuutena ampua vihollisaluksen lisäksi avaruudessa sinkoavia meteoreja tai ampua erityyppisillä aseilla. Useista pienistä asioista koostuu pelin ominaisuudet. Pelisuunnittelun alkuvaiheessa ominaisuuslistaa on vaikea suunnitella pitkälle sillä ne selkeytyvät kehityksen vaiheessa. Alkuun on kuitenkin hyvä listata pääasioita ja priorisoida niiden tärkeyttä. Kaikkia hyviä ideoita ominaisuuksiin ei kannata heti käyttää. Julkaisemalla jo julkaistuun peliin tasaisin väliajoin uusia ominaisuuksia voidaan myös pitää pelaajat koukutettuina pitempään. (21.)

3.14 Pelin koukut

Pelin koukuilla tarkoitetaan asioita jotka tekevät pelistä sellaisen, että pelaaja palaa aina uudelleen pelin äärelle. Pelin koukkuja voidaan huolellisesti suunnitella etukäteen mikäli niissä on pelin toiminnallisuuden kannalta oleellisia asioita joita on otettava huomioon suunnittelun kannalta tai ne voivat yllättäen suunnittelematta pelinkehityksen edetessä selvitä. (22, s. 45–46.)

3.15 Ansaintamalli

Ansaintamallia kannattaa myös etukäteen pohtia sillä se voi huomattavasti vaikuttaa pelisuunnitteluun. Yksinkertaisimpana ansaintamallina voi olla vain kerran ostettava peli ilman lisäkuluja. Pelaaja voi nauttia maksettuun hintaan pelin alusta loppuun. Mobiilipeleille ansaintamallit ovat suhteellisen yksinkertaisia. Esimerkiksi maksavaan peliin voi lisähinnalla ostaa lisäkenttiä tai vaihtoehtoisesti peli voi olla täysin ilmainen, mutta peliin voi ostaa pieniä lisäapuja joita kutsutaan myös mikromaksuiksi. Mikromaksut ansaintamallina voivat olla oiva keino saada suuremmat tulot pelistä. Mikromaksujen tuomat suuret tulot eivät kuitenkaan ole itsestäänselvyys vaan hinnoittelu täytyy olla tarkoin mietitty. Täytyy myös olla pelisuunnittelussa jo huomioitu mitä etuja ja koukkuja mikromaksut luovat. Ansaintamallin valintaan voi myös vaikuttaa markkinoilla olevat kilpailijoiden pelit. Mikäli markkinoilla on hyvin samantyyppinen peli kertaostos-tyyppisellä hinnoittelulla niin oma peli voisi olla kannattavaa suunnitella mikromaksupohjaiseksi peliksi.

4 TUOTANTOVAIHEET

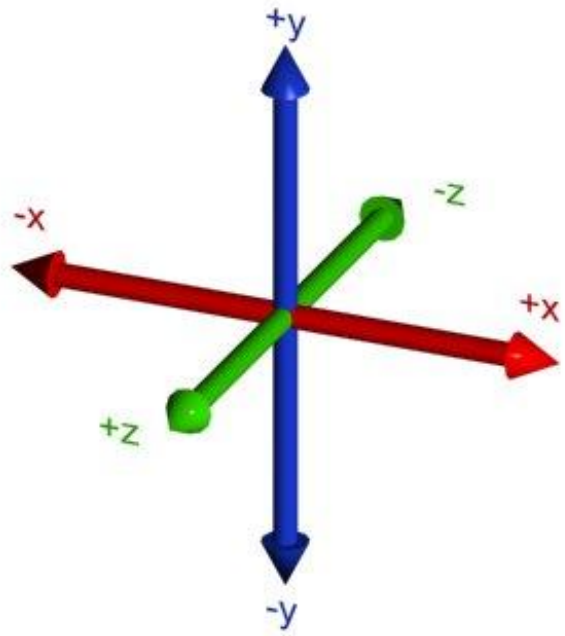
Tämän osion tarkoituksena on esitellä tuotantovaiheita workflown eli työnkulun kannalta, sekä valaista 3D-maailman saloja ja Unityn keskeisimmät perusasiat.

4.1 Kolmiulotteisuus

Ennen Unity3D:n pariin siirtymistä on hyvä ymmärtää perusasiat kolmiulotteisuudesta. 3D-pelien kolmiulotteisuudet noudattelevat matemaattisia ja fysikaalisia ilmiöitä.

4.1.1 Koordinaatit

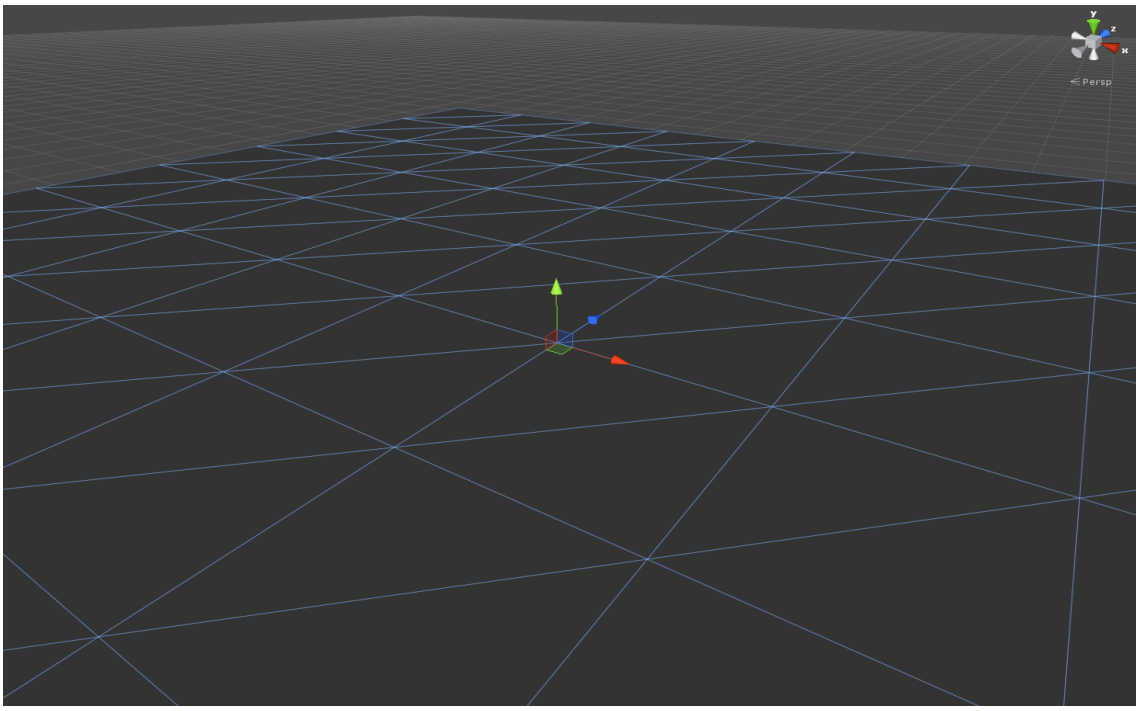
Kolmiulotteisesta tilaa kuvatessa käytetään Cartesian-koordinaatistoa. Cartesian-koordinaatistossa suunnat merkitään X-, Y-, Z-akseleihin eli leveys-, pituus- ja syvyyssuunnassa. Koordinaatit ilmoitetaan myös X, Y, Z-järjestyksessä ja merkitään laittamalla ne sulkeisiin järjestyksessä (x, y, z) (kuva 3). Samalla tavalla ne merkitään myös pelinkehityksessä. (24.)



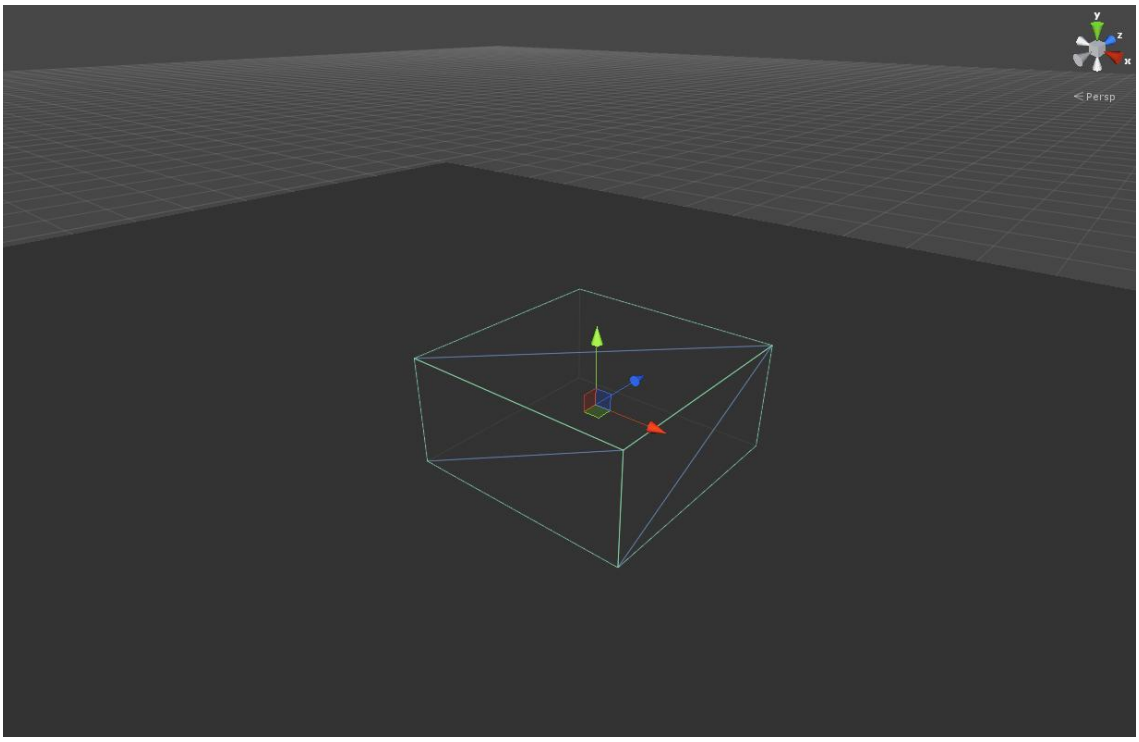
KUVA 3. Kolmiulotteinen karteesinen koordinaatisto(24.)

4.1.2 Avaruudet

Avaruudella tarkoitetaan yleisesti kolmiulotteista tilaa jolla ei ole rajoja, jossa sijaitsee objekteja ja jossa tapahtumilla on suhteellinen paikka ja suunta. Pelinkehityksessä kolmiulotteinen maailma luodaan tyhjään avaruuteen, joka sisältää peliobjektit. Avaruuksia tai toisin sanoen tiloja voidaan tarkastella erikseen maailma- (kuva 4) ja objekti-avaruuksina (kuva 5) sekä ne ovat suhteessa toisiinsa. Maailma-avaruuden keskipisteeksi on määritelty XYZ-akselin nollapiste, josta lasketaan objektien sijainnit tai etäisyydet muista objekteista. (25, s. 10.)



KUVA 4. Maailma-avaruus Unity3D-editorin Scene-näkymässä



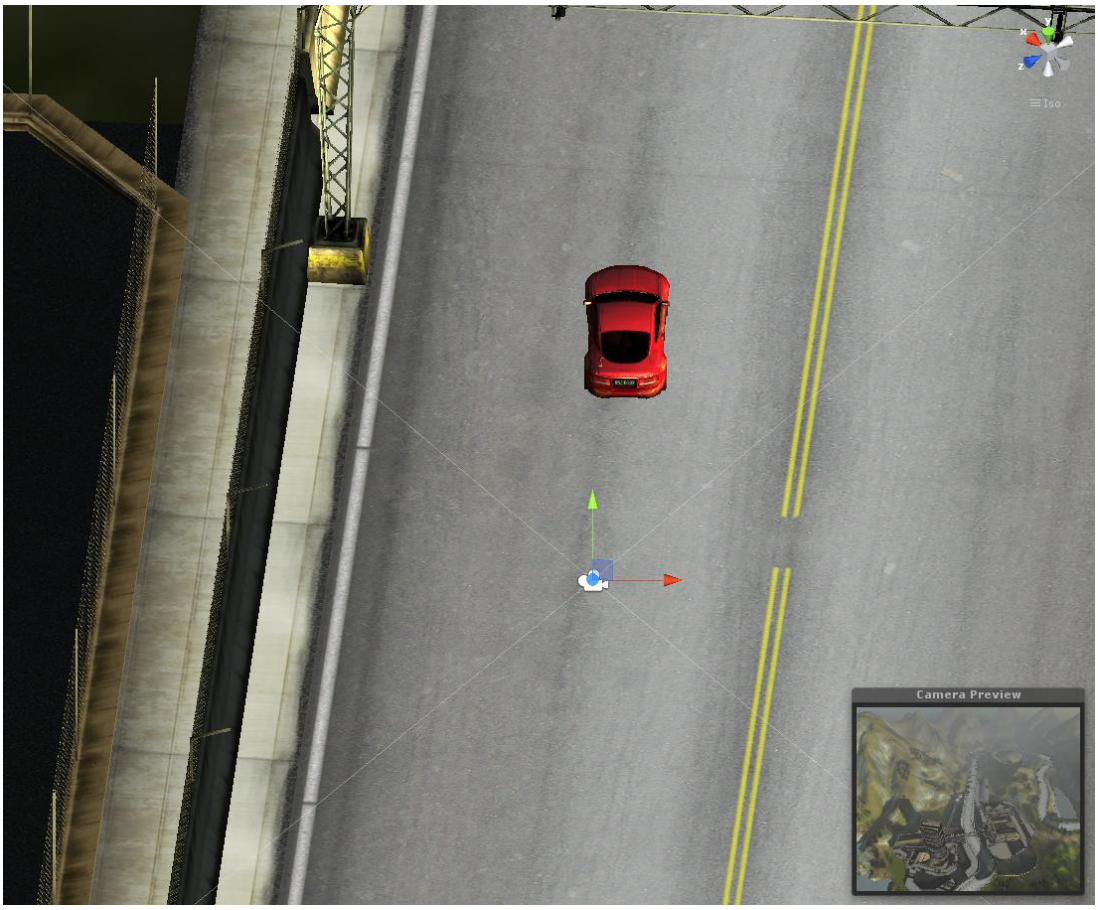
KUVA 5. Objekti-avaruus Unity3D-editorin Scene-näkymässä

4.1.3 Vektorit

Vektorit ovat yksinkertaisesti viivoja, jotka ovat piirrettynä kolmiulotteiseen maailmaan ja joilla on suunta ja pituus. Vektorit toimivat pelimaailman rakentamisessa samalla tavalla XYZ-koordinaatiston mukaisesti. Vektoreilla voidaan piirtää peliobjekteja ja niiden avulla voidaan laskea objektin pituuksia, kulmia sekä suunta. (25, s. 10).

4.1.4 Kamerat

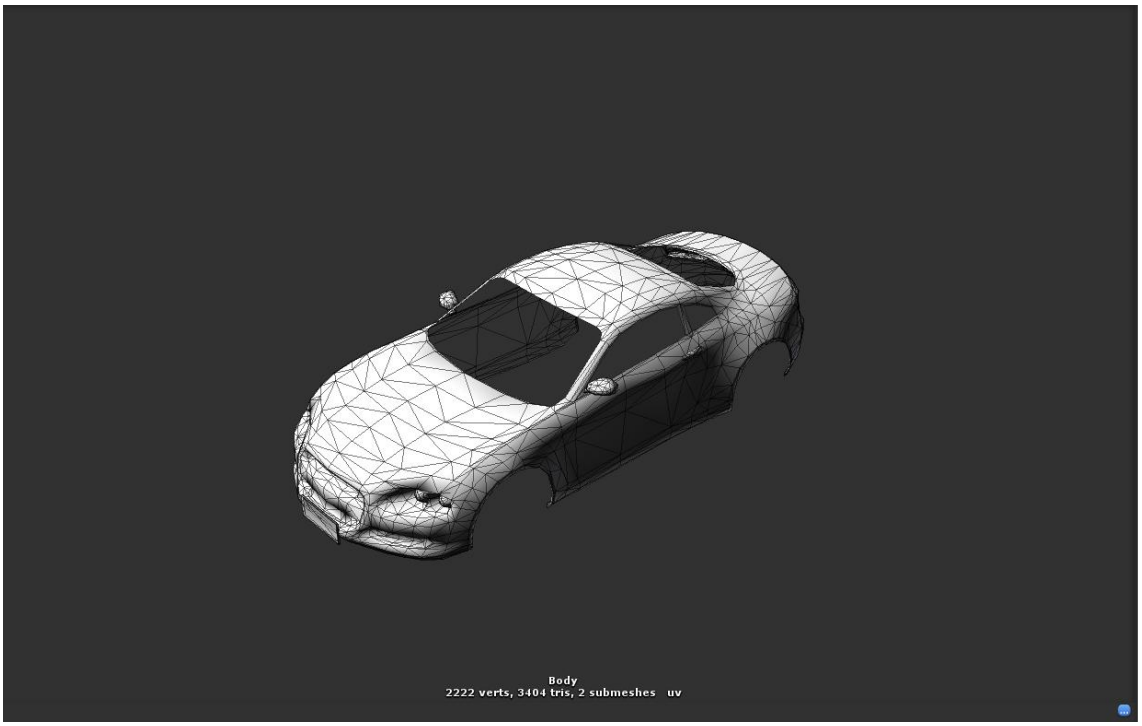
Kamera toimii kolmiulotteisessa maailmassa käytännössä samalla tavalla kuin ihmisellä silmät oikeassa maailmassa. Kameran tehtävä on avata pyramidimainen näkymä pelimaailmaan paikasta mihin se on asetettu (kuva 6) ja minne se on suunnattu. Kamera voi toimia osana pelihahmoa tai mitä tahansa peliobjektia jonka niin sanotuin silmin pelimaailma nähdään (25, s. 10–11).



KUVA 6. Kamera on asetettu auton yläpuolelle demo-esimerkissä

4.1.5 Monikulmiot, materiaalit ja varjostukset

Pelimaailmassa kolmiulotteiset kappaleet muodostuvat pienistä monikulmioista. Mitä enemmän monikulmioita on kappaleessa sitä tarkemmaksi saadaan kappaleen graafinen ulkoasu. On syytä huomioida, että mitä enemmän ja tiheämmin monikulmioita on, sitä enemmän se vaatii laitteelta suoritustehoa peliobjektien renderöintiin, joka tarkoittaa kuvan luomista ohjelmallisesta kuvausdatasta. Esimerkiksi mobiilipelejä tehtäessä on huomioitava mobiililaitteiden rajallinen teho. Siirrettäessä 3D-mallinnusohjelmalla tehtyjä modeleita Unityyn Unity muokkaa monikulmiot kolmioksi (kuva 7).



KUVA 7. Unityn demo-esimerkki -auton kori-modeli importattuna Unityyn

Monikulmiot yhdistyvät toisiinsa pisteinä kulmakohdista ja yhdistyneet monikulmiot muodostavat meshin eli verkkomaisen kolmiulotteisen muodon. Pisteiden tarkoituksena on olla kohtia joista voidaan laskea kappaleeseen mahdollisesti kohdistuvia törmäyksiä. Esimerkiksi voidaan selvittää ampumispelissä ammutun luodin tarkka osumakohta joka on tärkeä ominaisuus pelin kannalta.

Materiaaleilla on hyvin tärkeä osa kolmiulotteisissa peleissä, koska ne luovat ulkoasun kolmiulotteisille modeleille. Materiaalien avulla voidaan myös havainnollistaa esimerkiksi onko pelissä oleva esine puuta vai terästä tai voiko pelaaja kenties särkeä kyseistä objektia. Materiaalien siirtäminen Unityyn on helppoa. Unityn pelimoottori luo uudelleen kaikki tuodut 3D-modelit ja näin ollen 3D-mallinnusohjelmassa tehdyt mahdolliset tekstuurit ja varjostukset siirtyvät automaattisesti Unityyn. Materiaalit voi myös luoda tyhjästä käyttämällä kuvia tekstuureina ja varjostukset voi valita Unityn valmiista kirjastoista, kirjoittaa itse varjostusskriptit tai hyödyntää Unity-yhteisön luomia varjostuksia.

Materiaalitekstuureja voidaan luoda esimerkiksi Adoben Photoshopilla luomalla kuvatiedosto ja tekemällä kuvatiedostosta neliön muotoisia ja kuvioltaan itseään

toistavaa. Sen ansiosta kerran tehty kuviollisesti toistuva tiili voidaan toistaa esimerkiksi modelin pinnalla niin monta kertaa, että koko 3d-malli on vuorattu tekstuuritiilillä. Tekstuurikuvan resoluutio täytyy noudattaa kahden kerrointa, esimerkiksi 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 ja niin edelleen, jotta pelimoottori osaa asetella tekstuurit oikein. On jälleen syytä ottaa huomioon se, että mitä suurempia tekstuuritiedostot ovat resoluutiolta sitä enemmän laitteistolta vaaditaan tehoa. (25, s. 11–12.)

KUVA 8. Unity demo-esimerkki -auton korin tekstuurit luotuna UV-layoutiin

4.1.6 Kappaleiden fysiikka

Jotta peliin saataisiin simuloitua realistisuutta, on peliin lisättävä pelin oleellisimmille objekteille rigid body -komponentti. Fysiikkamoottorin tarkoituksena on simuloida fysikaalisia ilmiöitä, kuten massaa, painovoimaa, nopeutta ja kitkaa. Fysiikkamoottorin asettaminen objekteille kannattaa harkita tarkoin, sillä fysiikkamallinnusten prosessointi vaatii huomattavan määrän prosessointitehoa tietokoneen prosessorilta. Esimerkiksi vain pelissä ohjattavan pelihahmo-objektille on järkevää asettaa fysiikkamoottorin alaiset toiminnallisuudet. (25, s. 12–13.)

4.1.7 Törmäykset

Törmäykset ovat tärkeimpiä ominaisuuksia peleissä, sillä voidaan tutkia pelimaailman sisällä olevien objektien mahdollisia törmäyksiä. Miltä kuulostaisi jos pelissä voisi ajaa autolla päin seinää ja mennä siitä läpi kuin ei olisi seinää ollutkaan. Collider-komponentilla voidaan asettaa peliobjektille ikään kuin näkymättömän verkon ympärille, joka ilmoittaa tapahtuvasta törmäyksestä pelimoottorille ja pelimoottori käsittelee törmäyksen halutulla tavalla. Esimerkiksi autopelissä tämän avulla voidaan luoda törmäys mikäli auto ajaa seinään.

KUVA 9. Unityn demo-esimerkki -auton Mesh Colliderit näkyvissä

On myös olemassa collider-komponentin lisäksi mesh collider, jonka avulla voidaan tarkentaa törmäystä. Esimerkiksi jos pelissä auto törmää seinään pienellä vauhdilla auton oikea etukulma edellä, mesh colliderin avulla saadaan tietää tarkalleen mikä kohta peliobjektin meshistä on osunut seinään. Tämän avulla voimme saada aikaiseksi sen, että auton etukulma menee lyttyyn. (25, s. 13.)

4.2 Unityn peruselementit

Tämän luvun tarkoituksena on esitellä Unityn peruselementit, jotta lukijalle muodostuisi kuva miten ja millaisista osasista Unitylla tehdyt pelit muodostuvat. Unity on työnkulun kannalta loogisesti hyvin ja järkevästi toimiva. Unity antaa mahdollisuuden keskittyä olennaiseen, eli pelin tekemiseen ja ideoiden nopeaan toteuttamiseen kunhan on opetellut kuinka Unityllä tehdään asiat. Unityn mahtavuus perustuu sen peliobjektiorientoituneeseen konseptiin. Unityssä pystyy helposti pilkkomaan koko pelin yksittäisiin ja helposti hallittaviin objekteihin, jotka ovat tehty yksilöllisistä komponenttiosasista. Komponenttiosaset taas sisältävät muuttujia, joita muuttamalla voidaan hallita komponenttia tekemään peliobjektillemme mitä ikinä haluamme. Tekemällä yksilöllisiä objekteja ja lisäämällä niihin haluttuja toiminnallisuuksia se mahdollistaa progressiivisen tavan edetä pelinkehittämisessä ja pelin laajentaminen on käytännössä mahdollista loputtomiin. (26).

Esimerkiksi pomppivan pallon luominen Unityssä kuvastaa Unityn workflown yksinkertaisuutta. Jos Unityllä haluttaisiin luoda pomppiva pallo, aloitettaisiin se luomalla pallo Unityssä, eli luomalla uusi sprehe mesh-tyyppinen game object jonka voi luoda klikkaamalla Game Object -valikosta. Sen jälkeen pallolle voidaan antaa rigid body -komponentti valitsemalla luotu pallo ja Inspector-valikosta klikkaamalla Add Component, josta löytyy osio Physics ja sen alta valinta Rigidbody. Rigid bodyn lisääminen mahdollistaa pallon toimivan pelissä fysiikan lakien mukaisesti. Nyt pallo siis pelin käynnistyessä vain putoaa pelikenttään. Viimeisenä luodaan uusi Asset nimeltä Physic Material jolle annetaan Inspectorissa Bounciness-arvoksi 1. Sitten palloon raahataan hiirellä Project Viewissä näkyvä Physic Material, joka asettaa pallolle collider -komponentin. Vain muutamalla klikkauksella sai aikaiseksi pomppivan pallon. (25, s. 14.)

4.2.1 Assets

Assetit toimivat pelin resurssitiedostoina tai niin sanottuina rakennuspalikoina. Asseteiksi lasketaan kaikki pelin graaffiset ja visuaaliset elementit, kuten esimerkiksi tekstuurit, modelit ja äänet. (27.)

4.2.2 Game scenes

Game scenesillä tarkoitetaan pelissä ikään kuin näkymiä tai tasoja johon peli voidaan jaotella. Esimerkiksi ensimmäinen scene on peliä käynnistäessä pelin latausikkuna. Toinen ikkuna olisi esimerkiksi pelin alkuvalikko ja itse pelissä scenet voivat olla eri pelikenttiä joiden välillä peli suorittaa latauksen. Scenejen jaottelu auttaa jakamaan pelin latausaikoja, sekä pelinkehitysvaiheessa testaamaan pelin eri osia yksittäin. (28.)

4.2.3 Game objects

Kun jokin asset otetaan käyttöön pelimaailmassa niin tulee siitä game object eli peliobjekti. Assetin vieminen game sceneen asettaa Unity uudelle peliobjektille vähintään transform-komponentin. Transform-komponentti on välttämätön sillä se kertoo Unityn pelimoottorille uuden luodun peliobjektin paikan, suunnan ja koon XYZ-koordinaatistossa. Tämän alkuasetelman jälkeen peliobjektille voidaan asettaa toiminnallisuuksia ja tällä tavalla Unityssä aloitetaan rakentamaan pelimaailmaa. (29.)

4.2.4 Components

Komponentit ovat myös Unityssä tärkeitä rakennuspalikoita. Komponentteja on lukemattomia määriä ja niiden tarkoitus vaihtelee paljon. Komponentit on pääasiallisesti kuitenkin tarkoitettu luomaan peliobjekteille toiminnallisuuksia, määrittelemään ulkoasuja ja melkein mitä vaan voidaan ikinä kuvitella. Unityssä voi luoda itse komponentteja, joita kutsutaan myös skripteiksi tai käyttää Unityn yhteisön luomia komponentteja nopeuttamaan pelin tekemistä. Unityssä on kuitenkin olemassa peruskomponentit, kuten aiemmin mainittu rigid body ja lisäksi peruskomponentteihin lasketaan yksinkertaisemmatkin elementit kuten valot ja kamerat. (30.)

4.2.5 Scripts

Scripts eli suomeksi ilmaistuna skriptit, ovat erittäin oleellisia Unityn pelinkehityksessä. Unity käsittelee skripteja komponentteina, jotka sisältävät pelimaailman toiminallisuutta. Esimerkiksi skripteilla voidaan tehdä ohjattavan

hahmon kontrollit kuinka hahmolla voidaan liikkua ja mitä hahmo voi tehdä pelissä. Unityssä skriptien kirjoittamisessa on hyvänä puolena se ettei tarvitse varsinaisesti koskea Unityn pelimoottoriin saati muokata sitä vaan ainoastaan käyttää hyödykseen ja ohjelmoida pelimaailmaan toiminnallisuutta. (31.)

4.2.6 Prefabs

Prefabseillä eli niin sanotuilla valmiilla palikoilla tarkoitetaan Unityssä sellaisia peliobjekteja, jotka voidaan tallentaa assetiksi. Ideana prefabeissa on se, että peliobjektista luodaan sapluuna, jota voidaan käyttää useaan otteeseen pelin aikana. Prefabit helpottavat ja nopeuttavat huomattavasti pelinkehitystä mikäli peli sisältää jatkuvasti toistuvia asioita, jotka kuitenki voivat sisältävää komponentteja ja erinäisiä asetuksia. Prefabien hyvä puoli on myös se, että siitä voidaan luoda yksilöllisiä instansseja jotka mahdollistavat niiden yksilöllisen muokkaamisen. Prefab toimii samalla periaatteella kuin ohjelmoinnissa luokasta luotu uusi olio. (32; 25, s. 16.)

Kuvitellaan esimerkiksi jos on luotu tasohyppelypeliin laatikko, jonka särkemällä voi saada avustuksia pelissä ja laatikko-objektille on annettu esimerkiksi massa ja kirjoitettu skripti sen tuhoutumista varten, on erittäin todenäköistä, että tarvitset laatikkoa moneen kertaan. Lisäksi hyvänä puolena prefabeissä on se, että niitä voidaan hyödyntää myöhemmin myös muissa peliprojekteissa. (32; 25, s. 16.)

4.3 Työnkulku

Pelinkehityksessä oikeanlaisen workflown eli työnkulun löytäminen on erittäin tärkeää. Oikeanlainen workflow auttaa hahmottamaan koko kehitysprosessia ja näin ollen helpottaa suunnitelmallista työskentelyä. Kuten sanonta kuuluu: hyvin suunniteltu on puoliksi tehty. Hyvällä työnkululla ja iteroivalla kehitysmenetelmillä saadaan aikaa säästävä, mutta tehokas tapa saada tulosta aikaiseksi. Työnkulkutapoja voi pelinkehityksessä olla yhtä montaa kuin on pelistudioita. Itselleen ja pelinkehitystiimilleen sopivimmat tavat työskennellä löytävät vain tekemällä ja kokemuksen kautta.

4.3.1 Esivalmistelut

Pelinkehittämisessä toteuttamisvaihe aloitetaan esivalmisteluilla. Esivalmisteluihin tulisi sisältyä jo valmis pelisuunnitelma ja selkeät konseptit pelimaailmasta ja sen sisälle sijoittuvista peliobjekteista, sillä modelien tekemisen aloittaminen on lähes mahdotonta ilman konsepteja tai suunnitelmia. Kun modelit on saatu riittävän valmiiksi voidaan modelien päälle suunnitella ja toteuttaa sen mukaan tekstuurit. Modelit on syytä tehdä ensimmäisten joukossa sillä niistä rakennetaan pelimaailma, tehdään peliobjektit ja niiden myötä pelin toiminnallisuus. Modeleita voidaan peliin joutua tekemään suuria määriä, joten on hyvä palata pelisuunnitelman pariin ja pohtia etukäteen esimerkiksi mistä osista koostuvat ensimmäiset pelitasot ja sen mukaan luoda modeleita, varsinkin jos pelinkehitystiimissä on erikseen tekijät modeleille ja ohjelmoimiselle. Tällä tavalla saadaan pelin eri osien luominen ikään kuin telaketjumaiseksi.

Vaihtoehtoisesti useamman henkilön pelinkehitystiimissä työtahti on huomattavasti vikkelämpi jaettujen roolien vuoksi. Tarkkojen ja yksityiskohtaisten modelien tekeminen on aikaa vievää puuhaa mutta tiimin kaikilla tekijöillä täytyy kuitenkin olla tekemistä jatkuvasti. Näin ollen valmiiden modelien sijaan voidaan käyttää yksinkertaisia niin sanottuja dummy-modeleita. Ne ovat erittäin nopeasti sinne päin -tehtyjä tikku-ukkojen tyylisiä modeleita. Dummyja käytetään hyväksi siten, että niillä jo tasosuunnittelijat mallintavat pelitasoja ja lisäksi ohjelmoijat voivat ohjelmoida peliin toiminallisuuksia vaikka varsinaiset modelit eivät olisikaan valmiina.

4.3.2 Assetit ja pelitasot

Käytettävät modelit tuodaan Unity-peliprojektiin assetteina. Tämän jälkeen on hyvä aloittaa pelimaailman ja scenejen rakentaminen. Suositeltavaa on, että assetteja tuodaan projektiin sitä mukaan kuin niitä tarvitaan. Lisäksi mitä kannattaa kategorisoida luomalla alikansioita eri asseteille selkeyttämään projektia. Ilman kunnollista kategorisointia projektin tiedostohierarkiasta voi tulla hyvin sekava, mikä voi aiheuttaa produktiivisen työskentelyn hidastumista, joten se on syytä ottaa huomioon heti projektin alkuvaiheessa. Pelitasojen

rakentamisvaiheessa kannattaa muistaa keskittyä itse tasojen ja maailman rakentamiseen, jonka jälkeen vasta pelitasoihin luodaan toiminnallisuutta.

4.3.3 Pelikontrollit

Kun peliin on luotuna pelimaailmaa riittävästi, voidaan siirtyä tekemään pelin kontrolleja. Pelikontrolleilla tarkoitaan niitä pääohjausominaisuuksia pelissä, joilla pelissä liikutaan tai ohjataan pelin päähahmoa tai vaihtoehtoisesti esimerkiksi yksinkertaisessa mobiilipelissä ammutaan sormella pyyhkäisemällä ritsan avulla vihaisia lintuja kuten Angry Birdsissä. Loogisesti ajateltuna on oltava ensin pelikenttä ennen kuin pelimaailmassa voi liikkua ja toimia.

4.3.4 Peliobjektit

Pelitasojen ja pelikontrollien jälkeen on peliin lisäiltävä itse peliobjektit. Esimerkiksi Angry Birdsissä peliobjekteina toimivat sortuvat rakennelmat, possut ja tietysti ritsa ja ammuttava lintu. Kaikkiin peliobjekteihin on lisätty fysiikkamoottorin alaiset toiminnalisuudet. Kun ritsalla ammuttu lintu saa tietyn liikemäärän, se lentää fysiikan lakien mukaisesti ja putoaa kohti maata vetovoiman saattelemana. Mikäli ammuttu lintu osuu rakennelmaan voi se mahdollisesti kriittiseen paikkaan osuessaan aiheuttaa rakennelman sortumisen ja sortuvan rakennelman palaset voivat tuhota possut niihin osuessaan.

4.3.5 GUI menu

Kun peliin on tehty pelattavia tasoja on syytä peliin lisätä graaffinen käyttöliittymävalikko. Valikon tarkoituksena on toimia aloitusruutuna pelille, josta pelaaja voi itse käynnistää pelin, mahdollisesti muuttaa asetuksia, katsoa piste-ennätyksiä, yhdistää pelinsä johonkin sosiaalisen median palveluun tai lopettaa pelin.

4.3.6 Viimeistelyt

Viimeistelyt on hyvä tehdä kehityskaaren loppupuolella. Viimeistelyihin voidaan laskea esimerkiksi äänien, musiikkien ja esimerkiksi pisteytyksen lisääminen peliin. Isossa kehitystiimissä jo aiemmin mainittu yksityiskohtaisten modelien aikaa vievä työ voi tarkoittaa sitä, että oikeat modelit ja tekstuurit asetetaan

peliin vasta loppupuolella. Sen jälkeen tehdään vielä viimeiset viilaukset pelikenttiin ja pelin toiminnallisuuksiin.

4.3.7 Testaus

Kun pelin valmistuminen häämöttää, on syytä hankkia pelilleen useampia testaajia pelinkehitystiimin ulkopuolelta. Näin voidaan saada neutraalia arviointia ja palautetta pelistä. Pelistä voidaan helposti myös löytää mahdolliset ohjelmointivirheet ja pelinkehityksen aikana tapahtuneet inhimilliset erehdykset. Testauksen yhteydessä kannattaa harkita palautteen keräämistä kirjallisesti ja tehdä siitä myös mahdollisimman kattava, jotta pelille voidaan muun muassa pohtia realistista hintaa sovelluskauppaan.

4.3.8 Tie sovelluskauppaan

Pelin valmistuessa ensimmäiseen versioonsa on peli saatava sovelluskauppaan, mistä ihmiset voivat helpoiten pelin ladata tai ostaa sekä ladata. Prosessi Unity-pelin saamiseksi Applen App Storeen ja Google Play -sovelluskauppaan on suhteellisen suoraviivainen.

Prosessi aloitetaan ensin kirjautumalla joko iTunes Connect -sivustolle mikäli peli on iOS alustalle tai Google Play Developer Console –sivulle jos kohdealusta on Android. Tässä vaiheessa täytyy olla ostettuna kehittäjälisenssi. Mikäli lisenssiä ei ole sen voi ostaa prosessin aikana. Sivustoilta löytyy painikkeet uuden sovelluksen lisäämiseksi. Painike avaa step-by-step-prosessin joka ohjaa täyttämään tarvittavat asiat sovelluksen kauppaan saamiseksi. Pelistä täytyy tehdä release-käännös halutulle alustalle ja käännetty peli lähetetään sovelluskaupan haltijalle prosessin aikana. Kun prosessi on saatu loppuun, jää peli sovelluskaupan ylläpitäjille tarkistettavaksi. Tähän voi kulua aikaa yhdestä kahteen viikkoon. (33; 34.)

5 POHDINTA

Opinnäytetyön tavoitteena oli tutkia vaatimuksia Unity3D-pelinkehitykseen aloittavaan peliyritykseen ja käydä läpi pelinkehityksen elinkaarta.

Aiheen valinta tuotti paljon pohdintoja. Aiheeni keksittyäni arvelin opinnäytetyöstä tulevan laaja, kuten siinä myös kävi. Opinnäytetyöhön olin alun perin suunnitellut teknistä toteutusta, joka olisi ollut demoversio iOS-alustalle suunnatusta mobiilipelistä. Työtä aloittaessani ja aiheeseen syvemmin perehtyessäni huomasin aihepiirin ja rajauksen olevan suurempi kuin alussa kuvittelin. Näin ollen päädyin tekemään täysin tutkimus- ja selvityspohjaisen työn. Huomasin ettei alunperin suunniteltu pelidemo olisi ollut niin olennainen osa varsinaista aihepiiriä ottaen huomioon internetistä ja kirjoista löytyvän huomattavan määrän ohjeita ja tutoriaaleja teknisiin toteutuksiin ja ratkaisuihin. Sen sijaan huomasin, ettei löytynyt sellaista materiaalia joka toimisi siltana uuden asian pariin, varsinkaan suomenkielisenä.

Aihe vaati huomattavan määrän aiheeseen perehtymistä. Luettavaa ja uuden opiskeltavaa oli huomattavat määrät sekä hyvien lähteiden etsiminen oli työlästä. Uusien asioiden prosessointi on erittäin aikaa vievää, sillä uutta asiaa on paljon ja se täytyy oppia ja sisäistää, jonka jälkeen voi vasta siirtyä kirjoittamaan olennaiset asiat tuottavasti ja ytimekkäästi.

Opinnäytetyön tekeminen antoi aivan uudenlaisen lähestymistavan uusiin asioihin. Opin muun muassa etsimään olennaisempia asioita valtavasta teoriamäärästä ja jäsentelemään olennaisimpia asioita. Asioita ei tullut päntättyä pelkästään, jonka jälkeen olisi riennetty jo kovaa kyytiä toteutuksen maailmaan, mikä on perin insinöörimäinen tapa.

Työn ja valmistumisen jälkeen tarkoituksena on aloittaa mobiilipelien tekemistä Unity3D-ympäristössä. Alkuun pelinkehitys tulee olemaan vielä sivutoimista ja aikaa kuluu uusien asioiden ja niksien opettelussa. Opinnäytetyö antoi myös hyvän teoriapohjan tulevaisuutta ajatellen.

LÄHTEET

1. Ohjelmistotuotanto. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Ohjelmistotuotanto. Hakupäivä 17.10.2012.

2. Ketterä ohjelmistokehitys. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Ketter%C3%A4_ohjelmistokehitys. Hakupäivä 18.10.2012.

3. Beck, Kent 1999. Extreme programming explained: embrace change. Indianapolis: Addison-Wesley Professional.

4. Scrum. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Scrum. Hakupäivä 23.10.2012.

5. Projektinhallinta. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Projektinhallinta. Hakupäivä 24.10.2012.

6. Unity 3d multi platform. 2012. Saatavissa: http://unity3d.com/unity/multiplatform/. Hakupäivä 31.10.2012.

7. Unity 3d for windows phone. 2013. Saatavissa: http://blogs.unity3d.com/2013/03/27/unity-4-beta-program-for-windows-phone-apps/. Hakupäivä 28.3.2012

8. Apple app store multiplayer support. 2012. Saatavissa: https://developer.apple.com/game-center/. Hakupäivä 1.11.2012.

9. Games made with unity. 2012. Saatavissa: http://unity3d.com/gallery/made-with-unity/game-list. Hakupäivä 14.11.2012.

10. Unity assets. 2012. Saatavissa: http://unity3d.com/unity/workflow/asset-workflow. Hakupäivä 16.12.2012.

11. Autodesk 3ds max features. 2013. Saatavissa: http://www.autodesk.com/products/autodesk-3ds-max/features. Hakupäivä 3.1.2013.

12. Blender 3d. 2013. Saatavissa: http://www.blender.org/. Hakupäivä 3.1.2013.

13. Gimp. 2013. Saatavissa: http://www.gimp.org/. Hakupäivä 4.1.2013.

14. Adobe photoshop education. 2013. Saatavissa: http://www.adobe.com/education/student-eligibility-guide.edu.html. Hakupäivä 4.1.2013.

15. Unity team license. 2013. Saatavissa: http://unity3d.com/unity/collaboration/. Hakupäivä 5.1.2013.

16. Github. 2013. Saatavissa: https://github.com/. Hakupäivä 5.1.2013.

17. Trello. 2013. Saatavissa: https://trello.com/tour. Hakupäivä 10.1.2013.

18. Unity license. 2013. Saatavissa: http://unity3d.com/unity/licenses. Hakupäivä 10.1.2013.

19. App store distribution. 2013. Saatavissa: http://developer.android.com/distribute/index.html. Hakupäivä 11.1.2013.

20. Google play transaction fee. 2013. Saatavissa: http://support.google.com/googleplay/android-developer/answer/112622?hl=en. Hakupäivä 11.1.2013.

21. Suppean pelisuunnitelman pohja. 2012. Saatavissa: http://www.ludocraft.com/pelisuunnittelija/Suppean_pelisuunnitelman_pohja.doc. Hakupäivä 3.11.2012.

22. Vuorela, Ville 2007. Pelintekijän käsikirja. Helsinki: BTJ Finland Oy.

23. Schell, Jesse 2008. Art of Game Design: A Book of Lenses. Burlington: Elsevier Inc.

24. Koordinaatisto. 2013. Saatavissa: http://fi.wikipedia.org/wiki/Koordinaatisto. Hakupäivä 4.2.2013.

25. Goldstone, Will 2009. Unity Game Development Essentials. Birmingham: Packt Publishing Ltd.

26. Unity workflow. 2013. Saatavissa: http://unity3d.com/unity/workflow/. Hakupäivä 14.2.2013.

27. Unity asset. 2013. Saatavissa: http://unity3d.com/unity/workflow/asset-workflow. Hakupäivä 15.2.2013.

28. Unity scene. 2013. Saatavissa: http://unity3d.com/unity/workflow/scene-building. Hakupäivä 18.2.2013.

29. Unity game objects. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/GameObjects.html. Hakupäivä 27.2.2013.

30. Unity components. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Components/Components.html. Hakupäivä 27.2.2013.

31. Unity scripting. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/Scripting.html. Hakupäivä 28.2.2013.

32. Unity prefabs. 2013. Saatavissa: http://docs.unity3d.com/Documentation/Manual/Prefabs.html. Hakupäivä 1.3.2013.

33. Unity how to submit to the ios app store. 2013. Saatavissa: http://unity3d.com/learn/tutorials/modules/beginner/platform-specific/how-to-submit-to-the-ios-app-store. Hakupäivä 25.3.2013.

34. Google play upload application. 2013. Saatavissa: http://support.google.com/googleplay/android-developer/answer/113469?hl=en. Hakupäivä 25.3.2013.